ARMA Modelling: A Simulation Case Study

Arun K. Tangirala

September 19, 2018

Building an ARMA model: Example

In this case study, we shall learn how to systematically build an ARMA model.

- We shall also take this opportunity to discuss other subtle aspects of time-series modelling.

Data generating process

Let us simulate an ARMA(1,1) process,

$$v[k] = \frac{1 + 0.6q^{-1}}{1 - 0.7q^{-1}} e[k]$$

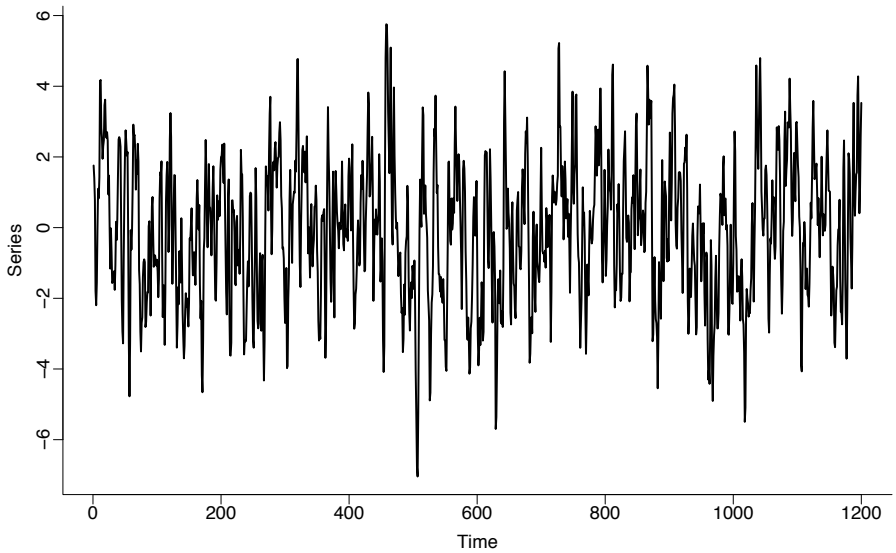
where $e[k]$ is a zero-mean, unit-variance GWN process.

```
# For purposes of illustration, generate WN for fixed seed
set.seed(240)  # can change it to any integer
N = 1200  # Length of series
ek = rnorm(N)
# Specify model and call arima.sim (look up help on arima.sim)
mod_dgp <- list(ar = 0.7, ma = 0.6, order = c(1, 0, 1))
vk <- arima.sim(mod_dgp, n = N, innov = ek)
```


Visualization

The first step is always to visualize the series and inspect it by the eye.

```
plot(vk, ylab = "Series", xlab = "Time", main = "", lwd = 2)
```



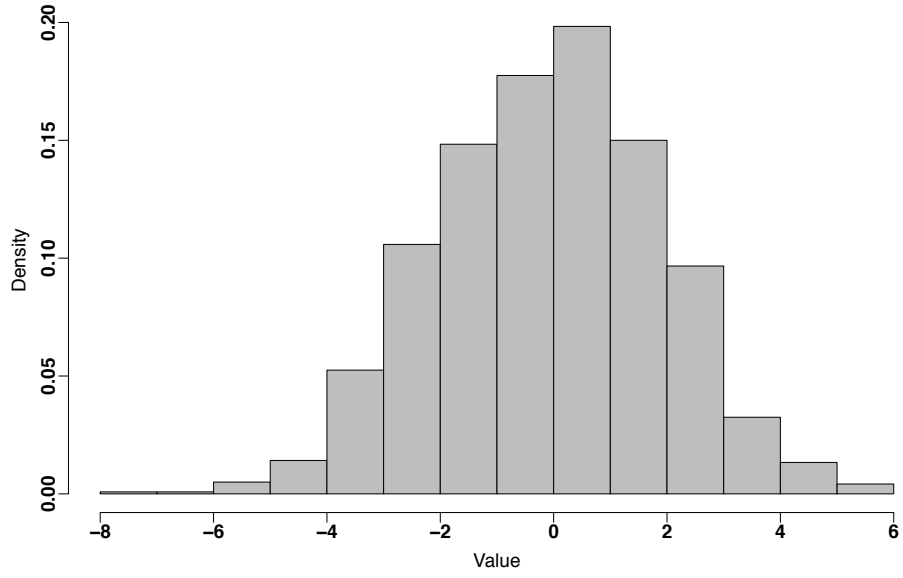
Inferences

- No visually evident non-stationarities
- Absence of outliers
- No missing observations

In a full-fledged modelling exercise, we should be conducting the stationarity tests to determine the presence of unit roots and/or trends. There exist efficient stationarity tests for this purpose (which we shall learn later).

Histogram

```
# Examine the histogram (to get a feel of the distribution)
hist(vk, probability = T, col = "gray", font.axis = 2, main = "",
    xlab = "Value")
```



Distribution estimate

The histogram is suggestive of a Gaussian distribution.

Q: - Which distribution have we examined? - Is it the marginal, joint? - sWhen is the histogram meaningful?

Note that we can turn to distribution fitting tools and obtain a rigorous estimate of the distribution or density function. However, the "visual" estimation should suffice for now.

It is customary next to examine the **summary statistics**

Summary

```
summary(vk)
```

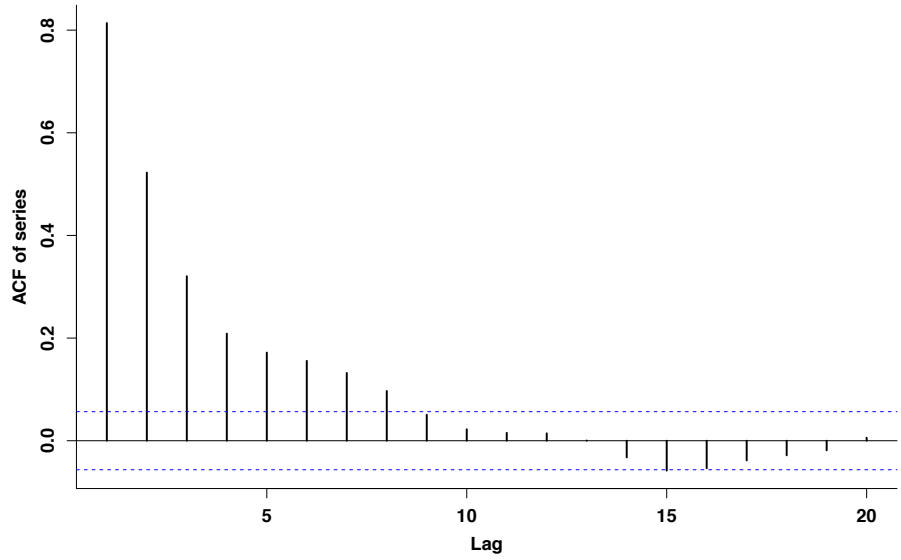
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -7.04169 -1.53187 -0.03349 -0.09634  1.33562  5.75367
```

Ideally one should conduct a test of zero mean at this stage (one-sample hypothesis test). Observe that the sample median and mean are close to each other (is this expected?)

Let's examine the auto- and partial auto-correlation functions next to determine the correlation structure.

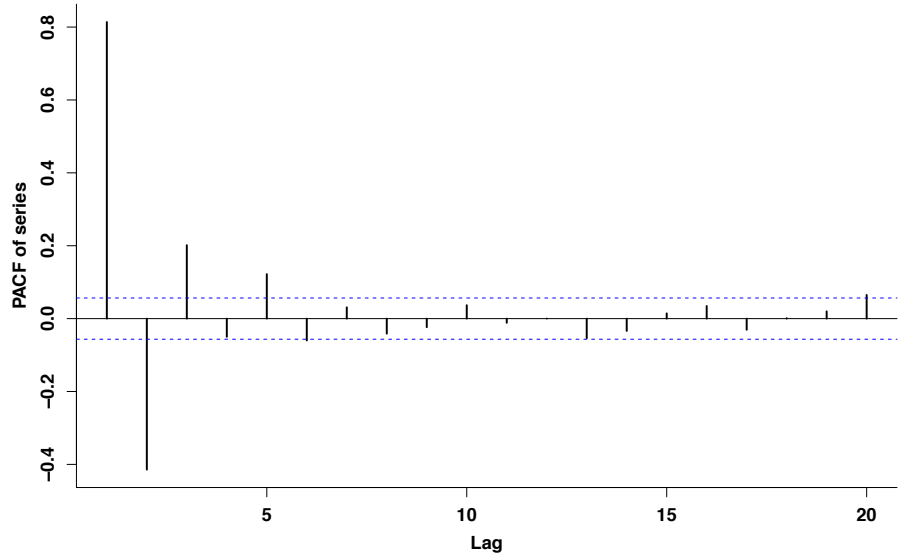
ACF of the series

```
acf(vk, lag.max = 20, ylab = "ACF of series", main = "", lwd = 2)
```



PACF of the series

```
pacf(vk, lag.max = 20, ylab = "PACF of series", main = "", lwd = 2)
```



Inferences on the correlation structure

- Series has significant temporal correlation - therefore a time-series model can be built
- Stationary process (rigorous tests are in general required)
- Moving average and auto-regressive behaviour observed.

What are the possible model structures and their respective orders?

Choices and general guidelines

We have three choices for the given series

1. Moving Average model of order $M = 8$
2. Auto-Regressive model of order $P = 5$
3. ARMA model of orders to be determined.

In the absence of any compelling **end-use** reasons, the following criteria may be used to determine the suitability of model type and order.

- Principle of parsimony
- Ease of estimation (and implementation)

We shall, therefore, estimate AR and ARMA models of appropriate orders.

Fitting an AR model

Let's estimate an AR model of specified order (as suggested by PACF)

```
ar_ord <- params$ar_P
```

$$v[k] + \sum_{i=1}^{5} d_i v[k-i] = e[k]$$

AR models can be estimated using linear LS methods.

```
arPmod = ar(vk, aic = F, order.max = ar_ord)
```

- The ar routine can search over a range of model order and return the best one as per the Akaike Information Criterion (AIC)
- Setting AIC to 'FALSE' amounts to forcing ar to fit the supplied order

Examining the model

```
print(arPmod)
```

```
##
## Call:
## ar(x = vk, aic = F, order.max = ar_ord)
##
## Coefficients:
##       1        2        3        4        5
##  1.2507  -0.7107   0.3460  -0.2019   0.1221
##
## Order selected 5  sigma^2 estimated as  1.043
```

How good is this model? Two criteria.

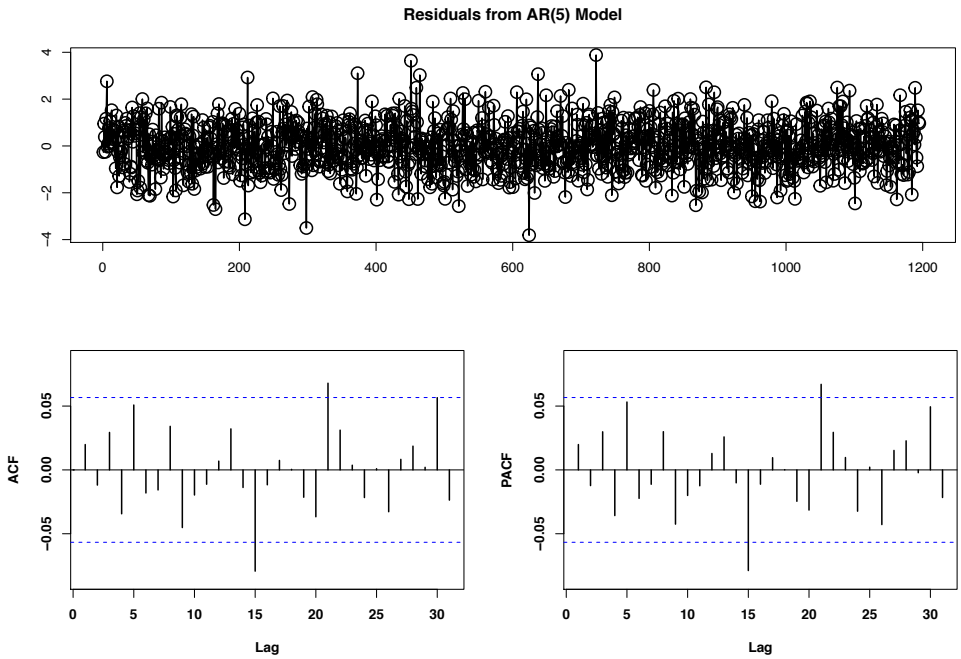
Model assessment

Goodness of model is evaluated in two respects:

- **Underfit:** Model should have captured the correlation structure adequately well. In other words, no bias in predictions on training data. This is tested via **residual analysis**, in particular, through the **whiteness tests**.
- **Overfit:** Model should not capture any realization-specific details. If it does, predictions on a fresh data will degrade. Alternatively, the variance in parameter estimates is higher than a sufficiently parametrized model. A **test of significance of parameter estimates** reveals overfits.

Residual analysis

```
title_str = paste("Residuals from AR(", params$ar_P, ") Model", sep = "")
tsdisplay(as.ts(arPmod$resid[ar_ord + 1:N]), main = title_str, font.lab = 2,
    cex = 2, font.main = 2, font.axis = 2, lwd = 1.5)
```



Inferences from residual analysis

- Residuals are white (based on the significance levels)
  - A more rigorous whiteness test is facilitated by the **Box-Ljung-Pierce (BLP) test.**
- Sufficient evidence exists, therefore, that the model has not underfit the data (process characteristics have been well-captured)

However, the risk is that the model could have overfit (modelling the realization specific characteristics)

Test for overfit: Significance tests of parameters

An implication of overfitting is that more parameters have been included than that could be identified from the given data. As a result, we obtain parameter estimates with "high" errors in the estimates. In order to detect this phenomenon, we conduct **hypothesis tests** (HT) of zero-truth on parameters.

Denoting the i^{th} parameter by θ_i, its true value by θ_{i0}, we are interested in the following:

$$H_0: \quad \theta_{i0} = 0$$
$$H_a: \quad \theta_{i0} \neq 0$$

Two-sided hypothesis tests of the above form are known as **significance tests**.

Significance tests

In order to conduct the aforementioned hypothesis test, what we have with us is the estimate $\hat{\theta}_i$. The term significance is used to essentially qualify the estimate $\hat{\theta}_i$ as being **statistically** small or large *given that the true value is zero*.

- If the latter is deemed to be true, then the null hypothesis $\theta_{i0} = 0$ is rejected, otherwise it fails to be. Hence the name given to these tests.

One of the best ways to conduct a HT is through the **confidence interval** approach.

Confidence intervals

The goal of any estimation exercise is not merely to arrive at a point estimate, but also to estimate the **region in which the truth resides.**

- In the presence of uncertainties, this region is always of non-zero width, which means we can never pin-point the truth (from finite sample sizes) with full confidence.
- Associated with any finite width region, is a degree of confidence (e.g., 95%). Hence the name.
- **Corollary:** The 100% confidence region for θ_0 has always infinite width (which is of course, of no use!). In fact, the confidence level and interval width requirements conflict with each other.

Note: Confidence intervals are constructed for true parameters and NOT for estimates.

Construction of CI

In order to construct confidence intervals (for any parameter), we require two pieces of information:

1. The estimate, $\hat{\theta}$.
2. Distribution (or density) of $\hat{\theta}$ including its form and first two moments (e.g., $\mu_{\hat{\theta}}$ and $\Sigma_{\hat{\theta}}$).

The procedure (of constructing CI) involves two steps. First, write the $100(1-\alpha)\%$ probabilistic interval (PI) for $\hat{\theta}$ using the given distribution. Second, from this PI, construct the $100(1-\alpha)\%$ CI for θ_0.

Two-sided HT using CI method

If the $100(1-\alpha)\%$ CI for θ_0 does not include the postulated value in the null H_0, then the we reject H_0 in favour of H_a at a significance level α.

Simple example

Example

Suppose $\hat{\theta}$ follows a Gaussian distribution

$$\hat{\theta} \sim f(\theta_0, \sigma_{\hat{\theta}})$$

and that we wish to construct 95% CI ($\alpha = 0.05$) for θ_0. Notice that we have an unbiased estimator (since $\mu_{\hat{\theta}} = \theta_0$).
Then, following the previously explained procedure, we have from the properties of Gaussian distributions:

$$\Pr\left(\theta_0 - 1.96\sigma_{\hat{\theta}} \leq \hat{\theta} \leq \theta_0 + 1.96\sigma_{\hat{\theta}}\right) = 0.95$$

$$\implies \theta_0 \in \left[\hat{\theta} - 1.96\sigma_{\hat{\theta}}, \hat{\theta} + 1.96\sigma_{\hat{\theta}}\right] \quad \text{(with 95\% confidence)} \quad (1)$$

Errors in estimates of AR parmeters

Returning to the modelling problem, the AR routine computes the estimates as well as the standard errors (in addition to several other quantities). To know the different values returned by the AR routine, execute in R

```
attributes(arPmod)
```

```
## $names
##  [1] "order"        "ar"           "var.pred"     "x.mean"
##  [5] "aic"          "n.used"       "n.obs"        "order.max"
##  [9] "partialacf"   "resid"        "method"       "series"
## [13] "frequency"    "call"         "asy.var.coef"
##
## $class
## [1] "ar"
```

The field `ar` contains the model coefficients, `resid` contains the residuals and so on. Of particular interest to us is the field `asy.var.coef`, which contains the asymptotic (large sample) variability in the estimates of AR coefficients.

Computing the standard errors and CI

We shall now compute the standard errors (in estimates) and hence the CI (for the true coefficients).

The field `asy.var.coef` contains $\Sigma_{\hat{\theta}}$, which is and $P \times P$ variance-covariance matrix of the vector $\hat{\boldsymbol{\theta}}$. Diagonals contain the variability in the individual estimates and off-diagonals contain $\sigma_{\hat{\theta}_i\hat{\theta}_j}$, i.e., the linear influence of error in $\hat{\theta}_i$ on that of $\hat{\theta}_j$. We shall be, generally, interested only in the diagonal elements to construct **approximate** confidence intervals.

The standard errors are computed as

```
# Examine the errors in estimates only if residuals are
# satisfactory
SigmaP <- arPmod$asy.var.coef
errPhat <- sqrt(diag(SigmaP))
```

Standard errors and CI

Examine the estimates and errors

```
cat("Estimates: \n", format(arPmod$ar, digits = 2), "\n")
```

```
## Estimates:
##   1.25 -0.71  0.35 -0.20  0.12
```

```
cat("Errors in estimates: \n", format(errPhat, digits = 2))
```

```
## Errors in estimates:
##  0.029 0.046 0.049 0.046 0.029
```

Given that $\hat{\theta}_i$ has an approximate Gaussian distribution, the 99% intervals are computed as

```
ci_coef <- cbind(arPmod$ar - 2.58 * errPhat, arPmod$ar + 2.58 * errPhat)
rownames(ci_coef) <- paste("P", 1:length(arPmod$ar), sep = "")
colnames(ci_coef) <- c("CILB", "CIUB")
```


CIs and refinement

Examining the confidence intervals

```
print(ci_coef, digits = 2)
```

```
##      CILB   CIUB
## P1  1.177  1.325
## P2 -0.829 -0.592
## P3  0.219  0.473
## P4 -0.320 -0.084
## P5  0.048  0.196
```

shows that none of the CI for θ_5 includes zero. Therefore, we reject the null hypothesis $\theta_{i,0} = 0,\ i = 1, \cdots, 5$.

- Note that the choice of model order, the coefficient estimates and therefore the CIs can change with the realization.

We now turn our attention to the ARMA model possibility and arrive at a suitable model using a similar procedure as above.

Fitting an ARMA model

The orders of the AR and MA component have to be determined by trial and error since no measure for an intelligent guess is available.

We shall start with ARMA(1,1) model:

$$v[k] + d_1 v[k-1] = e[k] + c_1 e[k-1]$$

- Estimaton of ARMA models requires the use of non-linear least squares (NLS) methods. Alternatively, the well-established maximum likelihood estimation (MLE) method can be used. In fact the MLE contains NLS as a special case.
- The `arima` routine in `R` is devised for estimating ARMA models (ARIMA is an extension of ARMA to include Integrating processes). By default it uses the MLE algorithm.

Fitting an ARMA model

We first fit an ARMA(1,1) model using the `arima` routine.

```
# Supply the series and order
arma11mod <- stats::arima(vk, order = c(1, 0, 1), include.mean = T)
# The second argument in the order corresponds to the integrating
# effect, which is assumed to be absent here.
```

Observe that we have asked the routine to estimate the mean (this translates to an intercept term in the model).

It is useful to know the fields of the model returned by `arima` routine.

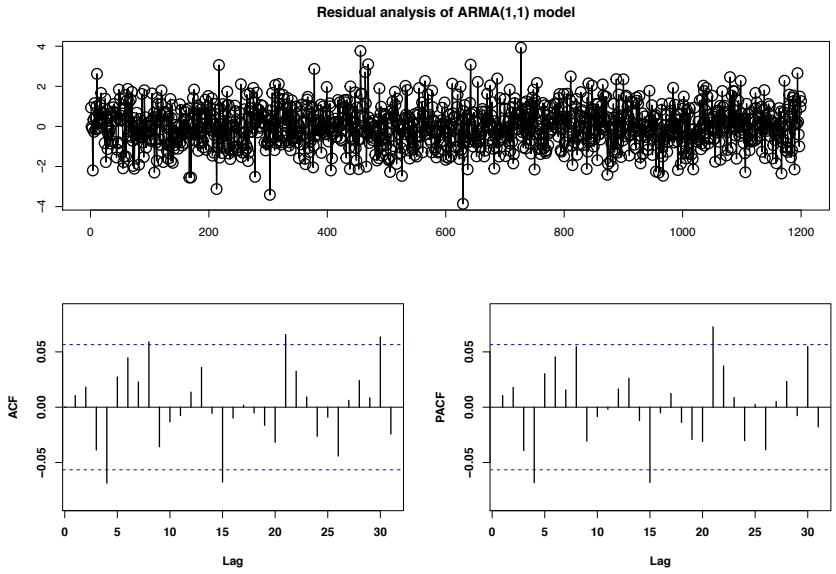
```
attributes(arma11mod)
```

```
## $names
##  [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"
##  [6] "aic"       "arma"      "residuals" "call"      "series"
## [11] "code"      "n.cond"    "nobs"      "model"
##
## $class
## [1] "Arima"
```

ARMA model assessment

We apply the usual model checks, residual analysis and significance tests for parameter estimates.

```
tsdisplay(as.ts(arma11mod$residuals), main = "Residual analysis of ARMA(1,1) model",
    font.lab = 2, cex = 2, font.main = 2, font.axis = 2, lwd = 1.5)
```



Significance tests

The `arima` routine reports the standard errors in the `var.coef` field. Furthermore, the estimates are asymptotically unbiased and follow a Gaussian distribution. Therefore, it is straightforward to compute the CIs.

```
ep_arma11 <- sqrt(diag(arma11mod$var.coef))
ci_coef3 <- cbind(arma11mod$coef - 2.58 * ep_arma11, arma11mod$coef +
    2.58 * ep_arma11)
pest_mat <- cbind(arma11mod$coef, ep_arma11, ci_coef3)
rownames(pest_mat) <- paste("P", 1:length(arma11mod$coef), sep = "")
colnames(pest_mat) <- c("Estimate", "Error", "CILB", "CIUB")
print(pest_mat, digits = 2)
```

```
##    Estimate Error  CILB CIUB
## P1    0.661 0.024  0.60 0.72
## P2    0.605 0.025  0.54 0.67
## P3   -0.084 0.138 -0.44 0.27
```

Significance tests

Observe that the estimate of the intercept term is insignificant. Therefore, we re-estimate the ARMA model omitting the intercept term.

```
armamod <- stats::arima(vk, order = c(1, 0, 1), include.mean = F)
ep_arma <- sqrt(diag(armamod$var.coef))
ci_coef4 <- cbind(armamod$coef - 2.58 * ep_arma, armamod$coef + 2.58 *
    ep_arma)
pest_mat <- cbind(armamod$coef, ep_arma, ci_coef4)
rownames(pest_mat) <- paste("P", 1:length(armamod$coef), sep = "")
colnames(pest_mat) <- c("Estimate", "Error", "CILB", "CIUB")
print(pest_mat, digits = 2)
```

```
##    Estimate Error CILB CIUB
## P1     0.66 0.024 0.60 0.72
## P2     0.60 0.025 0.54 0.67
```

Note: We have not carried out the mandated residual analysis. It is left as a homework.

Examining the AR and ARMA models

```
arPmod  # Summary does not work with models from 'ar'
```

```
##
## Call:
## ar(x = vk, aic = F, order.max = ar_ord)
##
## Coefficients:
##       1        2        3        4        5
##  1.2507  -0.7107   0.3460  -0.2019   0.1221
##
## Order selected 5  sigma^2 estimated as  1.043
```

```
armamod  # Summary works and provides additional details
```

```
##
## Call:
## stats::arima(x = vk, order = c(1, 0, 1), include.mean = F)
##
## Coefficients:
##          ar1     ma1
##       0.6618  0.6049
## s.e.  0.0239  0.0253
##
## sigma^2 estimated as 1.026:  log likelihood = -1719.16,  aic = 3444.32
```

Which model do we choose?

The final question now that we face is:

Which among the AR(5) and the ARMA(1,1) models is the more appropriate model for the given series?

Points to reflect

- AR(5) model has five parameters whereas the ARMA(1,1) model has two. Therefore, w.r.t. parsimony, the ARMA model is the winner.
- The estimates of AR(5) coefficients are unique whereas only local minima of ARMA(1,1) model estimates are obtained. In this respect, AR models are preferred.
- Goodness of fit, as indicated by the residual variance σ_ε^2 is lower for the ARMA model than the AR model. Therefore, ARMA is better choice.

Selecting the model

To answer the question formally, we can turn to **model selection criteria**. A widely used criterion is the Akaike Information Criterion (AIC) (others include Bayesian IC and Final Prediction Error criteria).

- The AIC is a quantitative measure of the trade-off between the approximation error (on the **training data**) and the variability in the parameter estimates (performance on a **test data set**).
- As the model complexity increases, the approximation error decreases, but the variability in $\hat{\theta}$ increases. The model with the lower AIC achieves the best trade-off and is the winner.

Akaike's Information Criterion

The AIC expression (as we shall learn later) contains two terms.

- The first term quantifies the model fit on the training data. It is proportional to the value of likelihood function achieved by the model.
- The second term measures the model's performance a test data. It is quantified by the dimension of the model, i.e., the number of parameters, and the sample size.

The AIC values for both models under consideration are returned by the `ar` and `arima` routines. However, the AIC from the `ar` uses a different calculation since it does not use the MLE method. In order to make a fair comparison, it is a good idea to use the MLE algorithm for both models.

Computing the AIC for both models

```
# Estimate AR model using MLE (for AIC computation)
armod_mle <- arima(vk, order = c(5, 0, 0), include.mean = F)
armod_mle  # Print the model
```

```
##
## Call:
## arima(x = vk, order = c(5, 0, 0), include.mean = F)
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ar5
##       1.2621  -0.7287  0.3663  -0.2226  0.1337
## s.e.  0.0286   0.0459  0.0494   0.0460  0.0287
##
## sigma^2 estimated as 1.021:  log likelihood = -1716.33,  aic = 3442.65
```

```
# Compare AICs of AR and ARMA models
armod_mle$aic
```

```
## [1] 3442.652
```

```
armamod$aic
```

```
## [1] 3444.321
```

Final model

Based on AIC, the AR(5) is better positioned than the ARMA(1,1) model. However, note the improvement in AIC *estimate* is negligibly small. Thus, we still choose the ARMA model over the AR model keeping in mind the economy of model complexity.

```
arma_th <- format(armamod$coef, digits = 2)
var_e <- format(armamod$sigma2, digits = 4)
ep_arma <- format(sqrt(diag(armamod$var.coef)), digits = 2)
```

The estimated model is:

$$v[k] = \frac{1 + \underset{(\pm 0.025)}{0.60}\, q^{-1}}{1 + \underset{(\pm 0.024)}{0.66}\, q^{-1}} e[k], \qquad e[k] \sim \mathcal{N}(0, 1.026) \tag{2}$$

- The model is in agreement with the data generating process (which is mostly a coincidence).
- **Cross-validation** on a test data set is left as a homework.